前言

簡介

無論你喜不喜歡，這個世界的開發模式已經不一樣了，它會隨著團隊甚至個人的不同而異。作為一位後端開發者，我瞭解這種急迫的需求：用真正可靠的資料庫來製作有效的身分驗證系統，讓你可以用它來保證安全，同時不會漏掉“管理誰可以操作哪塊區域”的身分驗證機制。

這種急迫性早就存在了，在行動優先的世界中，你一定要改善應用程式的安全性、維護可擴展的通知系統、以及製作優秀的使用者介面及體驗，所以改善行動 app 的功能是至關重要的任務，但即使完成上述的工作，你也有極重要的數據分析、創造收入的部分需要完成。大家都希望用簡單、輕鬆且無縫的方式來聆聽使用者的需求以及瞭解他們的不滿之處。

Firebase 用一組互聯的產品來提供上述的功能，可讓 app 輕鬆地擁有 Firebase 提供的一切。本書將許多問題 / 解決方案食譜分成十三章來探討各個主題，它介紹的主題都取自真實的案例，這些案例都是你開發新舊應用程式的過程中可能面對或即將面對的情況。

本書內容

第 1 章，初探 Firebase　說明將 Firebase 及其服務整合至各種平台與環境的程序，從前後端專案到 Android / iOS 專案。

第 2 章，Firebase Real Time 資料庫　介紹最常用的 Firebase 功能之一——Firebase Real Time，說明如何實作日常的資料輸入、取回以及更新，也介紹如何用更好的方式架構資料，最後介紹如何啟用所有的功能，以及離線啟用它們。

第 3 章，使用 Firebase 存儲來管理檔案　解釋如何用 Firebase 存儲來上傳、下載與管理檔案。

第 4 章，*Firebase* 身分驗證　介紹用 Firebase 來驗證使用者的各種方式，從傳統的身分驗證，到有異於 Facebook、Google 與 Twitter 的 OAuth 式登入程序。

第 5 章，使用 *Firebase* 規則來保護應用程式流程的安全　解釋如何用強大的 Firebase 授權規則來保護 Firebase 資料庫與 Firebase 存儲。

第 6 章，以 *Firebase* 實作漸進增強式 *App*　展示如何用服務工作與 Firebase 來將功能老舊乏味的 app 變成漸進增強式 app。

第 7 章，*Firebase Admin SDK*　說明如何建立基本的儀表板，以及和具備更多的授權方式且更強大 API 的 Firebase 功能互動，來管理使用者與通知。

第 8 章，用雲端功能擴展 *Firebase*　探討如何使用 Firebase 雲端功能與整合它，來與各種不同的 Firebase 產品互動，以擴展功能，並且在 Firebase 主控台內進行部署互動，來產生無伺服器架構。

第 9 章，完成後，我們來部署吧！　說明如何將程式部署到 Firebase Static 承載，並且使用組態設置來製作一些自訂的使用者體驗。

第 10 章，整合 *Firebase* 與 *NativeScript*　說明在許多平台上的 app 的 NativeScript 內使用 Firebase 的方式。

第 11 章，在本機整合 *Firebase* 與 *Android* / *iOS*　說明如何實作 Firebase 功能，包括與 Realtime 資料庫互動，以及在 Android 與 iOS 的原生環境中做身分驗證。

第 12 章，改造 *App*　說明一些可改善使用者體驗的小功能，包括邀請使用 app，以及發出符合訂閱主題的通知。

第 13 章，加入數據分析，將收益最大化　展示如何整合數據分析與 AdMob，用各種廣告來創造盈收。

附錄，*Firebase Cloud FireStore*　說明 Firebase Cloud Firestore 強大之處，以及它與之前的模型有哪些差異。

閱讀本書需要的工具

前十章的內容相當簡單，無論你使用哪種作業系統與程式編輯器都看得懂。

但是第 *11* 章，在本機整合 *Firebase* 與 *Android* / *iOS*，會開始開發行動 app，雖然你熟悉的是其他的作業系統（macOs、Linux 或 Windows），但是在開發 Android app 時，就要使用 macOS 電腦來跟著使用 iOS 的食譜一起操作。

本書對象

如果你想要在個人或公司專案中使用 Firebase 就適合閱讀這本書。本書包含眾多平台與各種開發環境，提供任何人都需要的所有知識。

我們只希望你真心願意瞭解為何 Firebase 這套互聯的工具組可以簡化開發者在建立專案或實作新功能時經常面對的問題。所以技術上來說，本書包含所有人想知道的所有內容。

編排方式

本書使用許多字體來表示各種不同的資訊。以下是這些字體的範例以及它們代表的意思。

在內文中，程式碼、資料表名稱、目錄名稱、檔名、副檔名、路徑名稱、虛擬 URL、使用者輸入，與 Twitter handle 的表示法是：“我們終於要寫 `put()` 方法了。”

程式區塊的表示法是：

```
// 取得檔案參考
 var rootRef = firebase.storage().ref();
 var imageRef = rootRef.child('images/<image-name>.
   <image-ext>');
```

新術語與**重要詞語**會用粗體字來表示。螢幕上的字詞，例如選單或對話框內的字詞，會這樣表示：“按下 **Upload to Firebase** 按鈕。”

1

初探 Firebase

本章將討論以下的主題：

- 建立第一個 Firebase app
- 將 Firebase 加入既有的前端專案
- 將 Firebase 整合後端
- 將 Firebase 整合到 Android app
- 將 Firebase 整合到 iOS app

簡介

當你想在劇烈變動的 web 與行動領域中尋找合適的解決方案與技術時，速度是必須考慮的要素。從 web 到行動開發，我們如何看待 API、資料與安全，以及如何盡可能地讓使用者參與其中，是很重要的課題。

傳統的設定（setup）到雲端都產生了許多新的模式與架構。**後端即服務（Backend-as-a-service (BaaS)）**可免除大量無用的設定與組態配置，讓我們只將注意力放在應用邏輯上。

接下來要介紹的 Firebase 是一種具備大量功能的 BaaS，可讓你輕而易舉地建立很棒的專案。它可以建立伺服器端程式碼並且提供更安全、精心構築的平台，免除許多繁瑣的工作甚至大量的人力，完全改變你對簡化與擴充性的看法。

所以，繫好安全帶，我們要從建立第一個 Firebase app 開始這趟旅程。

建立第一個 Firebase app

建立 Firebase app 的過程很簡單，這個過程大部分都是視覺化的。這個食譜將要展示從頭開始建立 Firebase app 的程序。

怎麼做…

1. 如前所述，我們只要使用功能強大的滑鼠與心愛的瀏覽器就可以開始工作了。先前往 Firebase 官方網站：`https://firebase.google.com/`。下面是這個網站的截圖（圖 1）：

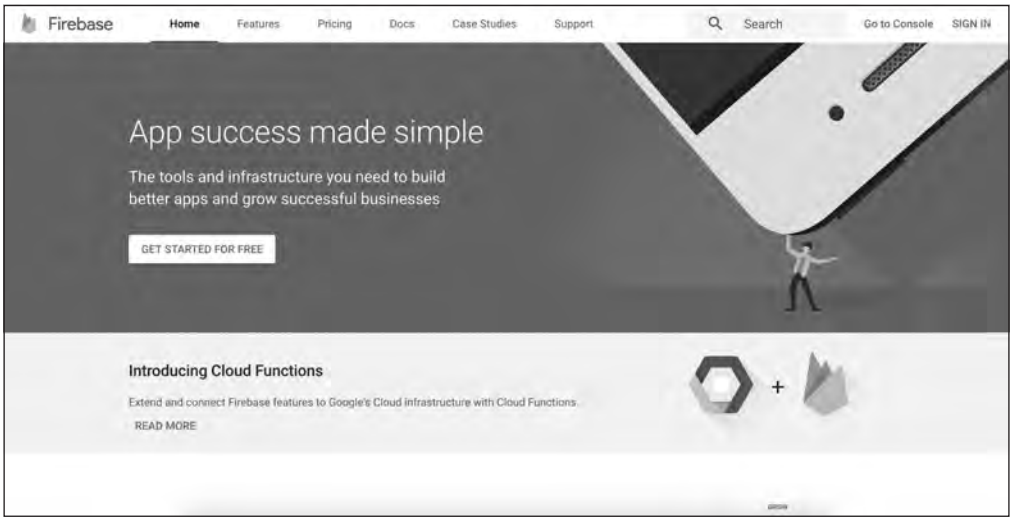

圖 1：Firebase 主畫面

2. 接著在導覽列按下 **SIGN IN** 按鈕。如下圖所示，你會看到 Google 的身分驗證網頁，並且可以在裡面選擇最適合進行開發工作的帳號（圖 2）：

圖 2：Firebase — Google 身分驗證

3. 選擇最適合的 Google 帳號後，你會被帶往這個連結：`https://console.firebase.google.com/`，裡面有所有的 Firebase 專案，你也可以加入新專案（圖 3）：

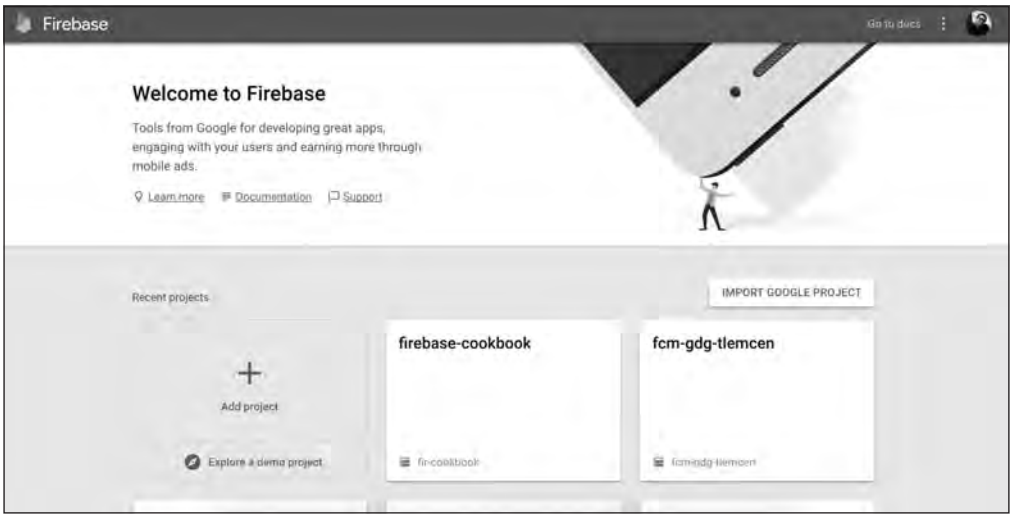

圖 3：Firebase 主控台

接下來我們有兩個選項：匯入 Google 專案，以及開始全新的專案。我們來瞭解如何建立新專案。

4. 按下**新增專案（Add project）**這個 + 號按鈕之後，你會看到一個填寫**專案名稱（Project name）**與**國家 / 地區（Country/region）**的畫面。請記得，**專案名稱**與**國家 / 地區**都是變數，所以你可以將它們的值改成適合你的值。下圖是建立專案的頁面（圖 4）：

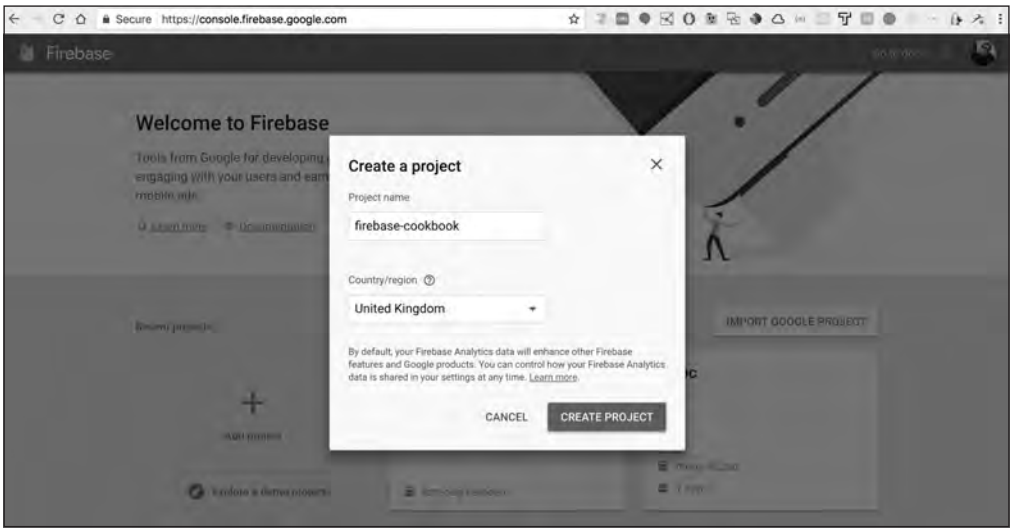

圖 4：建立 Firebase 專案

5. 完成上一個步驟之後，你會跳到 Firebase 儀表板（圖 5）：

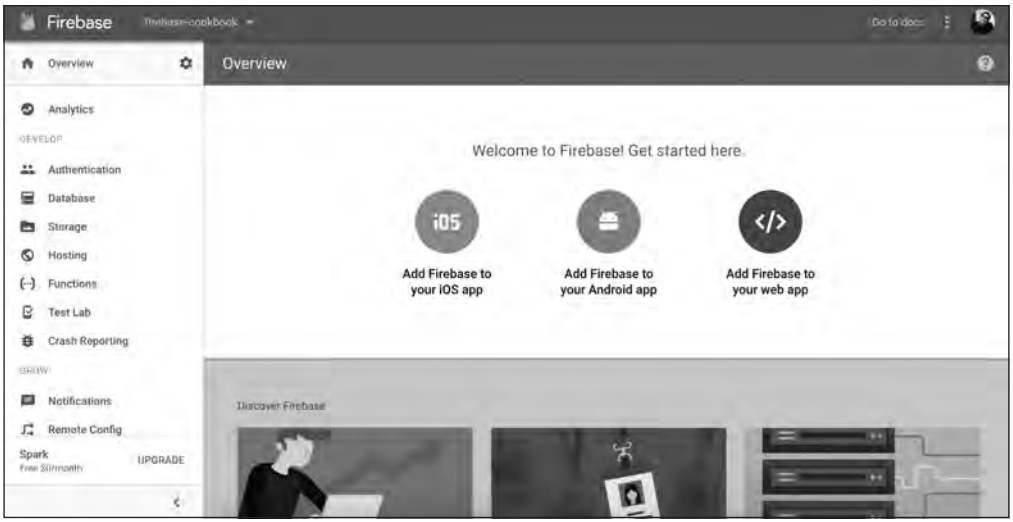

圖 5：Firebase 專案儀表板

恭喜！你已經成功建立第一個 Firebase 專案了。你可以看到這些步驟很簡單，而且可用於你現在或未來建立的任何 Firebase 專案。

將 Firebase 加入既有的前端專案

由於 Firebase 實際上是提供服務的後端平台，所以現在的開發人員不想要親手建立後端並不是件奇怪的事情。他們想把所有的注意力都放在前端，這也是現今無伺服器架構的主要構想。

怎麼做…

為了將 Firebase 完全整合到以 `.html`、`.css` 與 `.js` 組成的前端專案，我們要採取以下的步驟：

1. 打開你習慣使用的程式碼編輯器，輸入以下內容：

```
<script
  src="https://www.gstatic.com/firebasejs/3.9.0/firebase.js>
</script>
<script>
```

```
    // 初始化 Firebase
    // 待辦事項：換成你的專案的自訂程式碼片段
    var config = {
        apiKey: "<API_KEY>",
        authDomain: "<PROJECT_ID>.firebaseapp.com",
        databaseURL: "https://<DATABASE_NAME>.firebaseio.com",
        storageBucket: "<BUCKET>.appspot.com",
        messagingSenderId: "<SENDER_ID>",
    };
    firebase.initializeApp(config);
</script>
```

剛才做的事情是從 Firebase 的 CDN 匯入它的核心程式庫，並且使用 Firebase 提供的組態設置物件來將它初始化。

2. 接下來，我們要從 Firebase 專案儀表板抓取預先填寫的組態設置表單，步驟很簡單 — 登入你的 Firebase 專案（圖 6）：

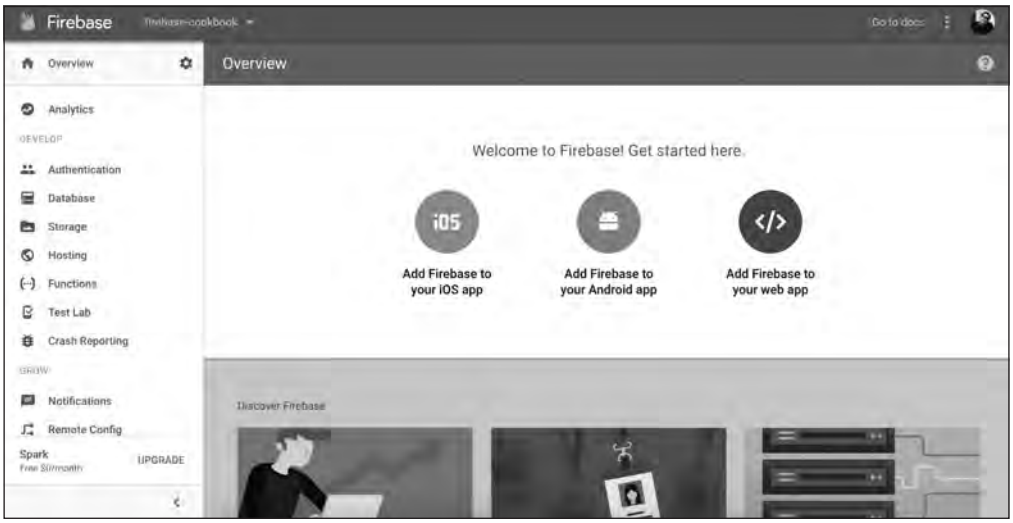

圖 6：Firebase 應用程式概觀 / 管理畫面

3. 按下洋紅色的按鈕 — 將 **Firebase 加入您的網路應用程式（Add Firebase to your web app）**，你會看到一個新畫面，裡面有所有必要的詮釋資料（metadata）（圖 7）：

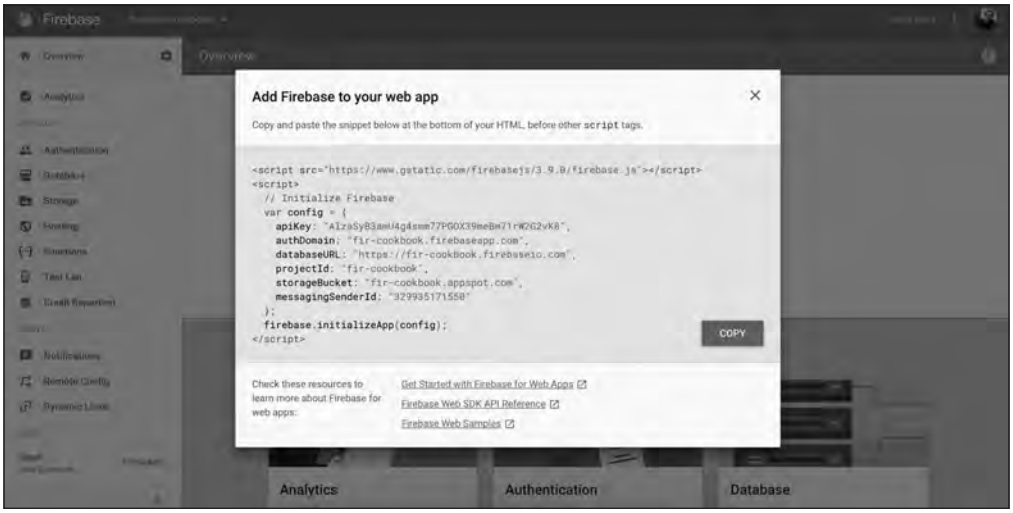

圖 7：Firebase 專案憑證

4. 複製畫面中的程式片段並貼到你的 `index.html` 網頁之後，即可關閉網頁。

恭喜你，你已經成功地將 Firebase 整合到 Firebase 專案裡面了。請記得，Firebase 的服務相當模組化，所以你不會有大量厚重的依賴項目，只需要使用一種（或最多四種）資源即可。

在下一個模組中，我們要瞭解如何將 Firebase 與後端 app 整合。

工作原理

我們在上一個步驟透過網頁來整合 Firebase JavaScript 用戶端，並且建立基本的骨幹組態配置。我們也按照文件的指示，複製並貼上存有必要的安全令牌（token）和 API 金鑰的組態腳本，之後 Firebase 需要用它們來支援我們的功能。

將 Firebase 整合後端

Firebase 是個可完全取代後端的完整解決方案，但是有時因為某些需求，你需要將 Firebase 整合至既有的後端。

此時，我們要在一個 NodeJS 後端應用程式中整合 Firebase 服務。

怎麼做…

因為我們使用 NodeJS，整合 Firebase 只需要做一個模組設定：

1. 直接在終端機（Windows 的 cmd）輸入下面的命令：

```
~ cd project-directory
~/project-directory ~> npm install firebase --save
```

   上面的命令會將 Firebase 下載到本地端，讓你可以用一般的 commonJS 工作流程直接使用它。

2. 接著要抓取 Firebase 專案的組態設置。這個步驟比較簡單，因為你可以採取上一節將 *Firebase* 加入既有的前端專案介紹的步驟來找到組態詮釋資料。

3. 用你喜歡的程式編輯器輸入這些內容：

```
// [*] 1: 在工作流程中 require 及匯入 Firebase
   const firebase = require('firebase');

// [*] 2: 用憑證來初始化 app
      var config = {
      apiKey: "<API_KEY>",
      authDomain: "<PROJECT_ID>.firebaseapp.com",
      databaseURL:
       "https://<DATABASE_NAME>.firebaseio.com",
      storageBucket: "<BUCKET>.appspot.com",
    };
  firebase.initializeApp(config);
```

恭喜！你已經成功地將 Firebase 整合到後端工作流程了。附帶一提，我們也可以使用一種叫做 Firebase Admin SDK 的東西來擴展既有的工作流程，第 7 章，*Firebase Admin SDK* 會討論如何整合以及使用它。

工作原理

類似前端整合，我們在後端做這些事情：

1. 使用 node package manager，也就是 npm 來安裝 Firebase `commonJS` 程式庫，它裡面有所有必要的 API。
2. require / 匯入 Firebase，讓它成為 app 的一部分，將它傳給一個組態物件，這個組態物件會保存所有的 API 金鑰、連結及其他資訊。
3. 最後使用剛才建立的組態物件來初始化應用程式。

將 Firebase 整合到 Android app

在 Android Studio 2.0 以上，Android Studio IDE 可讓你更輕鬆地使用 Firebase，所以整合 Firebase 各種元件是一種愉快的體驗。

準備工作

為了建立準 Firebase Android app，你必須在開發機器安裝 Android Studio，你可以到 `https://developer.android.com/studio/index.html` 下載適合你的開發機器作業系統的版本。

怎麼做…

成功下載 Android Studio 之後啟動它，你會看到下面的歡迎畫面（圖 8）：

圖 8：Android Studio 歡迎畫面

接下來要建立一個新的 app。這個程式很簡單：

1. 填寫應用程式名稱、類型與適用的 SDK 之後，你的 Android 應用程式開發工作流程是（圖 9）：

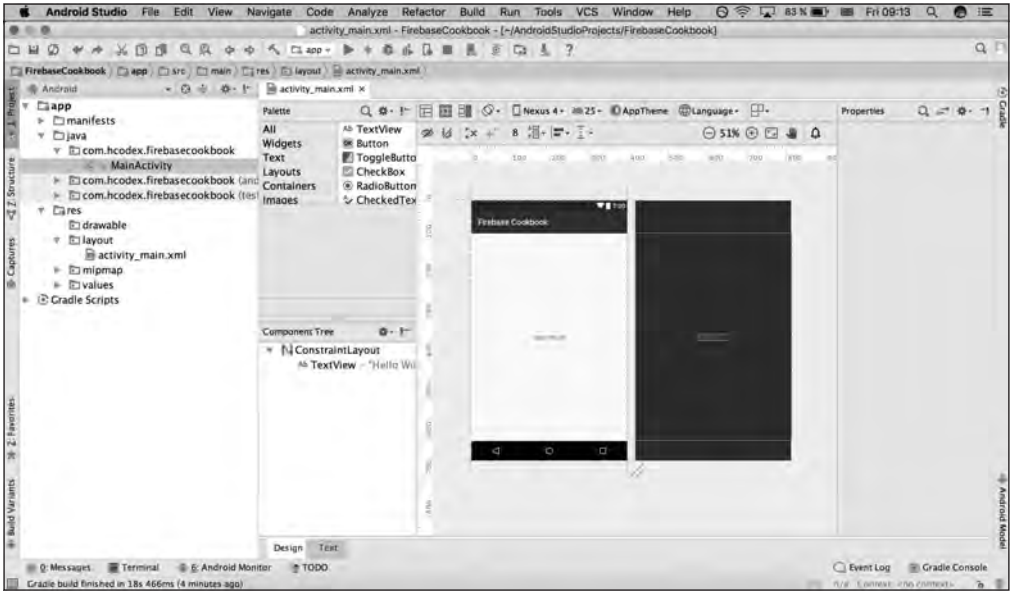

圖 9：啟動應用程式後的 Android Studio

2. 接下來很好玩。直接在 Android Studio 選單按下 **Tools** 選項後，你會看到一個項目裡面有 **Firebase** 等選項（圖 10）：

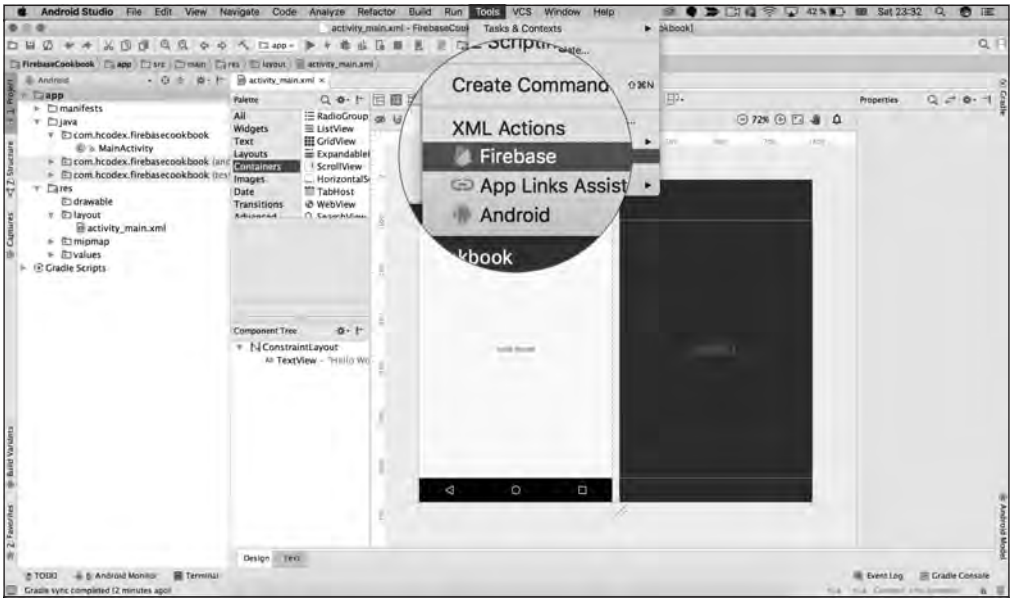

圖 10：將 Firebase 整合到 Android app

3. 按下它之後，你可以在右側的 **Assistant** 裡面找到 Firebase 部分，裡面有它提供的所有好東西（圖 11）：

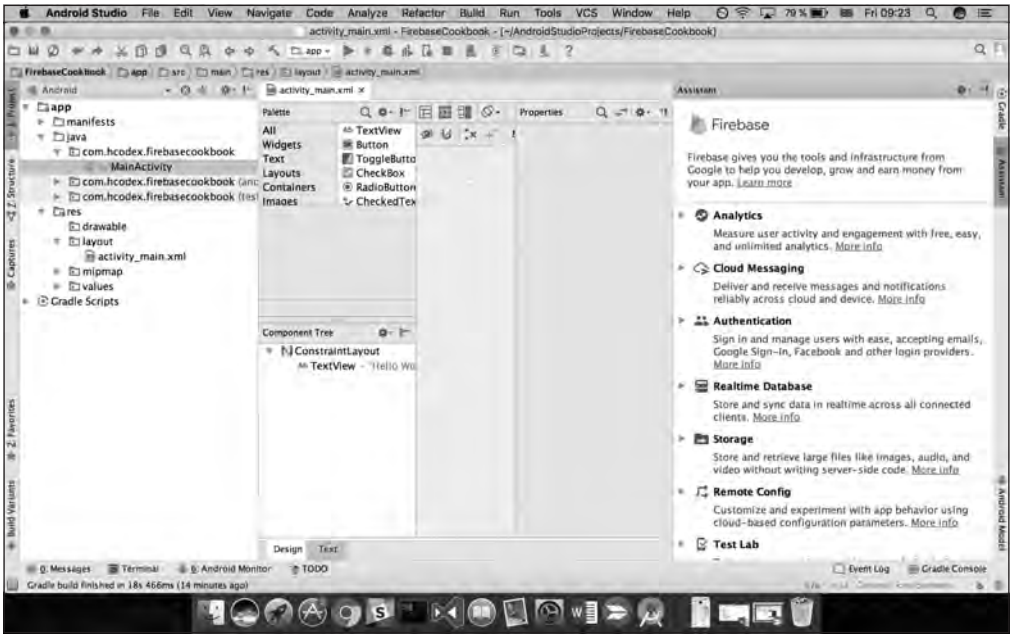

圖 11：整合 Android 與 Firebase — 第一部分

你可以在這個區域看到 Firebase 提供的所有東西 — 裡面有我們之前提過的各個部分與區域。Firebase 是許多服務的集合，也就是說，它的每個部分本身都是一種服務，這也代表你可以自由選擇想要加入的服務。本章將用 Realtime Database 來結束整合程序的說明。

各種 Firebase 服務都可以用完全相同方式來執行這個範例的整合程序。

4. 按下 **Realtime Database** 之後，你會看到一個副選單，裡面簡單解釋與說明該服務的實際功能（圖 12）：

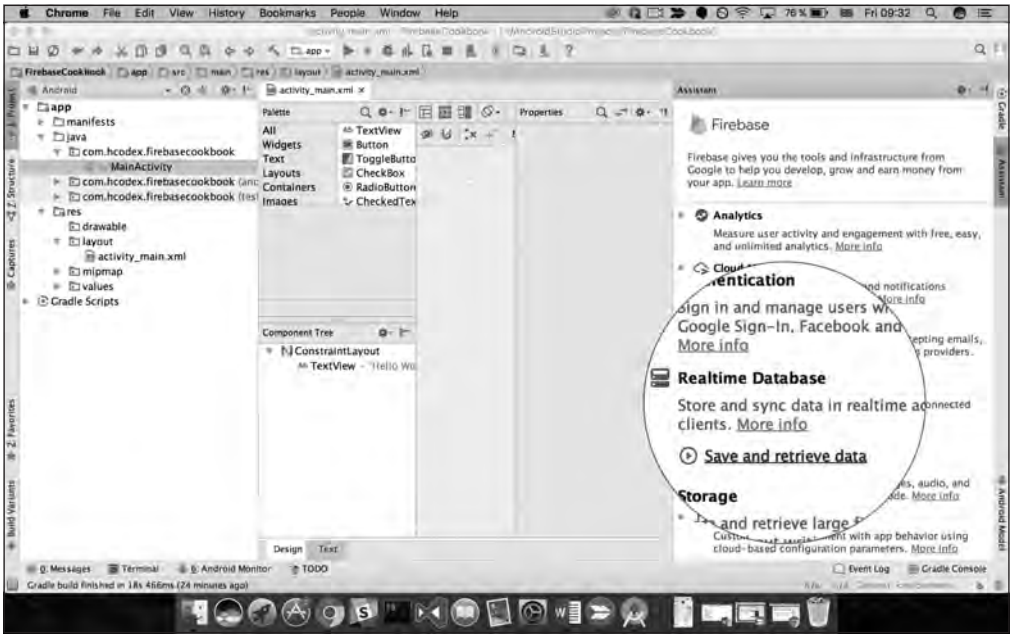

圖 12：整合 Android 與 Firebase — 第二部分

5. 接下來，你只要按下 **Save and retrieve data** 連結選項就可以啟動一個新程序，這個程序可做身分驗證，以及下載 Firebase 元件並安裝到你的 app（圖 13）：

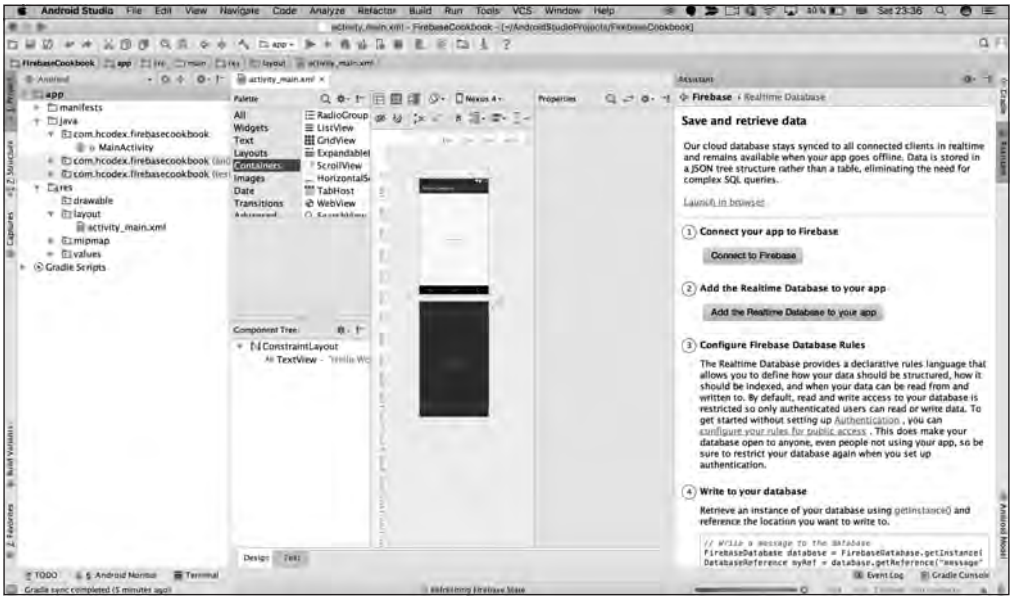

圖 13：Firebase 專案整合 — 第三部分

接著使用之前的做法來設置專案。接下來你要做身分驗證，使用你的 Firebase 專案所使用的 Gmail 帳號。

6. 按下連結之後，你要選擇專案的 Google 帳號。此時，你必須授權 Android Studio 使用你的 Google 帳號。當你批准那些授權規則之後，會看到下面的網頁（圖 14）：

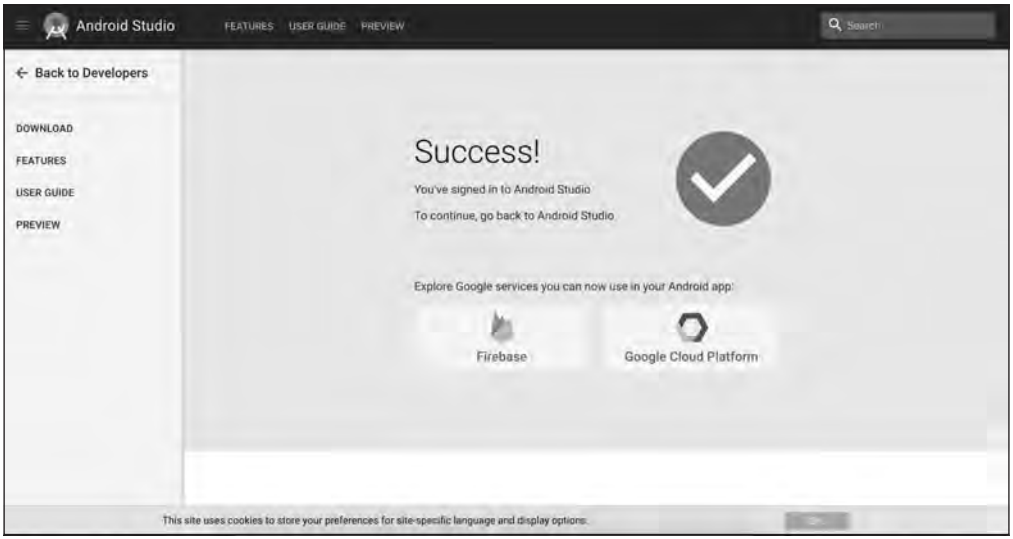

圖 14：Firebase 專案整合 — 第四部分

恭喜！現在 Android Studio 已經完全連接你的 Google 帳號了。此時你可以看到 Android Studio 跳出一個新畫面。如前所述，你可以選擇一個 Firebase 專案或建立一個新專案。在本例中，我們已經建立酷炫的專案了，所以只需要選擇它，並按下 **Connect to Firebase** 按鈕（圖 15）。

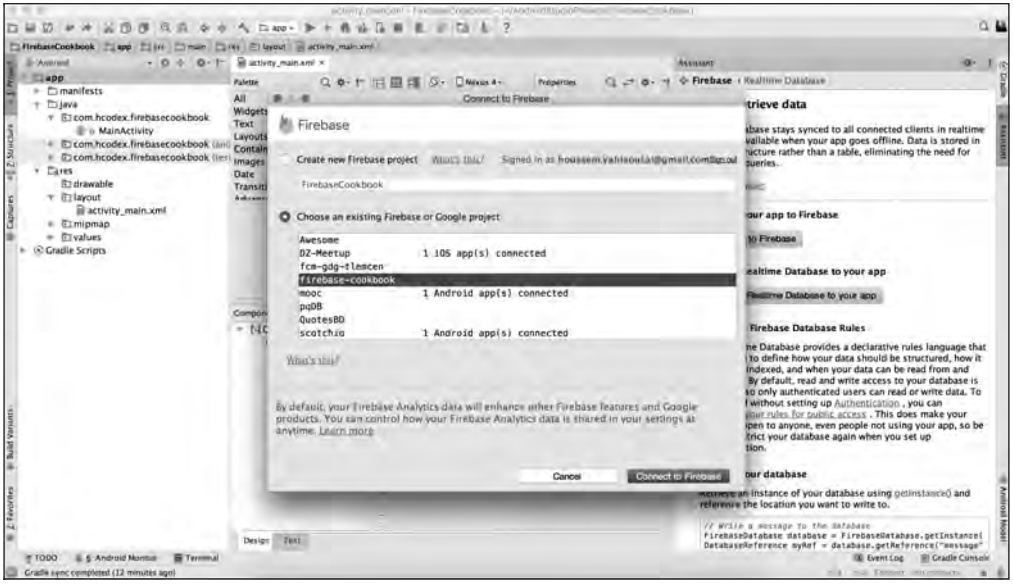

圖 15：Firebase 專案整合 — 第五部分

接下來 Android Studio 會花幾秒鐘來連接專案並設置酷炫的 app。接下來，你會看到下面這個可愛的綠色按鈕，代表一切事物都順暢地進行（圖 16）。

圖 16：Firebase 專案整合 — 第六部分

恭喜！現在你的 app 已經完全連結並且可以承載 Firebase 邏輯了，接下來只要整合你渴望擁有的服務，並直接使用它即可。第 *11* 章，在本機整合 *Firebase* 與 *Android / iOS* 將會開始介紹它們與其他資訊。